Hindcast Robot User's Manual

Version: HEFS 1.2.1

Release Date: 28 October 2014

National Weather Service
Office of Hydrologic Development

Contents

1 Benefits of Using the Hindcasting Robot

Using the Robot to run the hindcast offers some dramatic advantages. First of all, running hindcast is a frequent, tedious and sometimes repetitive task with fixed procedures. The run at a specific T0 does not depend on the results of running previous T0s. Very often, the whole hindcast period needs to be split into several short periods for various reasons:

• So that they can be run simultaneously with different SAs on several different computers to fully utilize the available CPUs.[1]

• In the past, it was observed that running a long period hindcast in one shot is a very bad practice because during the run the localDataStore size increases, which slows down the run.[2] In the past, each hindcast run period, using different SAs, was limited to only two years to avoid large size localDataStore. Currently this problem has been solved due to code change. The run is reported that does not slow down even after several years' run. However, it's still a good practice to pay attention to it.

• When the leap days to be skipped[3], for example, when day interval is 5 and the total run period is from 01-01-1985 12:00 GMT to 12-31-1998 12:00 GMT, dividing the whole period into 4, or even finer, is necessary:

01-01-1985 → 02-28-1988;
03-02-1988 → 02-28-1992;
03-02-1992 → 02-28-1996;
03-02-1996 → 12-31-1998

This divide and conquer technique dramatically increases the work load for the user and has a greater chance of a typo type of error. In contrast, using the Robot releases these types of burdens. For each run, an input text file containing Start T0 and End T0 and other information needs to be created. Then the user can feed these input files to the robots. Besides running in the parallel mode, the Robot can execute the run tasks in the series mode where several runs are in a queue. Anytime one run is finished, the next run is started automatically.

Using the Robot has other additional advantages, compared to running by a human:

• Sometimes, for unclear reasons, FEWS skips one or two days in the hindcast run. At the end of the run, the Robot checks and re-runs the missed days if that occurs.[4]

[1] In today's world, computers usually have multiple CPUs. Our experiment shows that running on a four CPU linux computer, the best way to fully utilize all CPUs is to split the whole period into four evenly divided sub-periods. Running the four SAs simultaneously on the same computer takes about half time than running a single SA with full period. Of course, running the four SAs on four different Linux computers takes 1/4 time of running single SA on a single computer with the whole period. Another experiment of ours shows a single 120 hours (5 days) hindcast run, if divided into multiple runs and executed on multiple different computers could be finished in one night.

[2] By checking the generated result files' time stamps, a user can find out how many minutes it took to process one T0.

[3] When day interval is greater than 1 and CFSv2 data is used for hindcasting, due to that CFSv2 data is not available on leap days, they need to be skipped during the hindcast run. Please see HindcastingGuide.docx section 3.3 "CFSv2 and Leap Days".

- The Robot detects certain types of errors: when a warm state was not found, FEWS uses a cold state silently. Users may not notice it at all, but the Robot will find it and flag it.

[4] Sometimes, a big block of days are without result files due to other reasons. If there are more than 10 days without result files, Robot will flag it and just quit, since this is not the case of random skipping.

2 Set Up

2.1 Software Background

The Hindcast Robot program is written in Java. It uses several technologies, including 3rd party Java program Sikuli and its dependent C programs (tesseract-OCR, openCV and other programs).

The Java programs run cross platform. All the related C programs have been compiled on our Linux computers into the shared dynamic object library files (*.so files). All the *.so files are provided. Once you have set the environment variable LD_LIBRARY_PATH correctly, shown below, there is no need to install any of these programs on your computers.

2.2 Verify Prerequisites

After untarring the HEFS release tar file, please find the directory *hindcastRobot*. As an example for this user manual, assume it is **/home/joe/hefs/hindcastRobot/**, and it should contain the following:

```
fews_hindcast_robot.jar
sikuliLibs/
        gcc_4.7.3/
        leptonica_1.6.9/
        opencv_2.4.1/
        tesseract-ocr_3.0.2/
wmctrl
```

2.3 Set LD_LIBRARY_PATH

Set the Linux environment variable LD_LIBRARY_PATH so that the shared object library files can be found and used by the computer programs:

```
sikuliDynamicLib=/home/joe/hefs/hindcastRobot/sikuliLibs (edit this line  based on
your  location)
export LD_LIBRARY_PATH=$sikuliDynamicLib/gcc_4.7.3:$LD_LIBRARY_PATH
export
LD_LIBRARY_PATH=$sikuliDynamicLib/leptonica_1.6.9:$LD_LIBRARY_PATH
export
LD_LIBRARY_PATH=$sikuliDynamicLib/opencv_2.4.1:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$sikuliDynamicLib/tesseractocr_3.0.2:
$LD_LIBRARY_PATH
```

To avoid typing these commands every time, you may copy these commands and put them in your **.profile** or **.bashrc** file, etc. based on the SHELL you are using and run it when you log in.

2.4 Set Path

Set the PATH so that the executable command wmctrl can be found and called:

```
export PATH=/home/joe/hefs/hindcastRobot/:$PATH
```

2.5 Check Java version on your computer

The Hindcast Robot program needs java 1.7 or newer to run[5]. You may check the version of your default java command with the following command:

```
java -version
```

If it is older than version 1.7, you need to update the Java executable. Currently, FEWS is running Java 1.7; therefore the jre/ used to run FEWS already contains one java executable (jre/bin/java) of version 1.7.0_11. You may use that copy of the executable if you want. For convenience (so that you don't need to type the long pathname each time), you may create an alias like:

alias java="/awips/chps_share/jre/bin/java" (edit it based on your computer)

Again, you may insert that alias line into your **.profile** or **.bashrc** file, etc., based on the SHELL you are using.

2.6 SA Configuration

The Robot will just run the default work flow, using the default forecast length. So the users need to make sure the SA configuration is ready:
i)The default workflow is the first one within Config/RegionConfigFiles/WorkflowDescriptors.xml
ii)The default forecast length is defined in Config/DisplayConfigFiles/ManualForecastDisplay.xml.
e.g. the forecast length of 15 days:

```
<forecastLength unit="day" multiplier="15"/>
```

[5] If you are not a Java developer and you don't have Java SDK installed on your path, you just need Java jre; i.e. Java run time environment, to run the Java program.

3 Run Hindcast by Using the Robot

3.1 Input text file

For running a hindcast using the Robot, we need to prepare the input text file(s). Each input text file specifies which SA to run, which FEWS script to use, what START T0 and END T0 are etc. Following is one example:

```
SA_FOLDER=/awips/hefshome/joe/hindcast/cnrfc_sa_job1
EXPORT_FOLDER=/awips/hefshome/joe/hindcast/cnrfc_sa_job1/Export/
FEWS_SCRIPT=/awips/hefshome/joe/hindcast/bin/fews.sh.rboff
START_T0=01-01-1987 12:00:00
END_T0=12-31-1987 12:00:00
FILE_NUM_PER_DAY=15
DAY_INTERVAL=..     (optional)
SCRIPT1=..     (optional)
SCRIPT2=..     (optional)
DEBUG=1  (optional)
```

The property names (the names on the left side of '=' sign) are fixed. The order of them does not matter. Following are more details:

- **SA_FOLDER** defines the SA for the Robot to run, and assumes all the configurations have been finished, see section 2.6 for reference.
- **EXPORT_FOLDER** and **FILE_NUM PER_DAY**: **EXPORT_FOLDER** define the parent directory, and/or subdirectories, containing the results files. At the end of the hindcast run, the Robot counts the number of the result files for each day to see if it equals to the number defined by **FILE_NUM PER_DAY** property. If no files were created, or fewer than expected/defined for that specific day, this day is considered to be skipped during the run. The Robot prints all the missed days on the terminal. If the missed days are fewer than 10, the Robot will re-run FEWS one more time for these missed days - one day, one FEWS run. However, if more than 10 days are skipped, it is probable that the cause is something other than random skipping, so the Robot will flag it and quit.

  Users have to find the value of **FILE _NUM PER_DAY**, the total number of files each day, on their own.
- **FEWS_SCRIPT** defines the script to run FEWS. You may run it with **fews.sh** or **fews.sh.rboff**.
- **START_T0** and **END_T0**: the format must be **MM-DD-YYYY HH:00:00**. **START_T0** must be before **END_T0**.
- **DAY_INTERVAL** default value is 1.
- Both **SCRIPT1** and **SCRIPT2** are optional. If present, the shell script(s) defined will be executed when the hindcast run is finished. So users may insert their own tasks into the Robot program, e.g. a script to send an email to the user to notify the run has finished, or compress the result files to minimize the disk space used. The difference between **SCRIPT1** and **SCRIPT2** is that the Robot waits for **SCRIPT1** to finish, but moves on while **SCRIPT2** is still running.

- The line containing **DEBUG** is also optional, and the default value is 0. When it is present and the value is 1, some debug messages will be printed out, helping to diagnose problems as they occur.

One such input text file is needed for one FEWS run. Assuming the whole hindcast run is split into several runs, several such files would be needed. Part of the content of these files is the same (e.g. FEWS_SCRIPT and FILE _NUM PER _DAY), and part of the content varies (e.g. START_T0 and END_T0). Comparing to manually entering those START T0, END T0 on several FEWS GUI, it is much less error prone to create/edit several text files and feed them to the Robot.

3.2 Run Hindcast

When all the input text files have been created, now we can run a FEWS hindcast by using the Robot. Running from an xterm, you may start one robot with the following command:[6],[7]

```
java -jar /p-a-t-h/fews_hindcast_robot.jar /p../input1.txt [/p../input2.txt ...]
```

The Robot will do the following steps automatically:

1. Launch FEWS against the specific SA using the specific script defined in the input text file;
2. Click on the Manual forecast button;
3. Select batch mode;
4. Enter Start T0 and End T0;
5. If DAY INTERVAL is specified in input text file and its value is not the default value 1, the Robot will set the interval to that specific value;
6. Select "User defined" radio button;[8]
7. Click Run button to start the run;
8. Open the logs panel for the user to visually inspect;
9. After 30 seconds, minimize the FEWS GUI as a shade.
10. After the run has reached to the end, the Robot will close its corresponding FEWS automatically and check if all the result files under the defined Export directory have been created for each day from Start T0 to End T0. If any day's results files are missing, the Robot will print that out on the terminal and re-start FEWS to run on that specific day. However, if there are more than 10 days with missing files, it is probable that the cause is not random skipping, so the Robot will flag it and quit.

[6] The command without parameters:

```
java –jar /p-a-t-h/fews_hindcast_robot.jar
```

prints helping information.

[7] The command with "scan" as the last parameter:

```
java –jar /p../fews_hindcast_robot.jar /p../input1.txt [..] scan
```

will not run hindcast. It just scans the result files under Export directory defined in the input text files and report if any days' result files are missing.

[8] The Robot does not set the forecast length! Users need to set it in the configuration file, DisplayConfigFiles/ManualForecastDisplay.xml as example of forecast length of 15 days:

```
< forecastLength unit="day" multiplier="15"/>
```

If there are multiple input text files feeding the robot, when the first run defined by input1.txt is finished and re-run (if needed), the second run defined by input2.txt will be started automatically, without human input required.

You may also run several robots in parallel mode.

In term 1:

```
java -jar /p../fews_hindcast_robot.jar /p../input1.txt /p../input2.txt
```

In xterm 2 (maybe on a different machine):

```
java -jar  /p../fews_hindcast_robot.jar /p../input3.txt /p../input4.txt
```

In xterm 3 (as above)

Notes:

- Each run of the Robot, a temporary helping directory called **”libs”** will be created by Sikuli in the containing directory of **fews_hindcast_robot.jar**. If you don’t have the ‘w’ permission to the containing directory, the Robot will fail.
  After the Robot has finished, you may delete this helping directory, **”libs”**, or just leave it.[9]

- Each robot needs to use the computer keyboard and mouse to enter information on its FEWS GUI. When one robot has control of the computer, it creates an empty file~~,~~ **”fews_hindcast_robot_running”,** in the user’s home directory, preventing other robots(if any) from starting their FEWS and using the keyboard and mouse. The dominant robot enters information on its FEWS GUI, clicks the ’Run’ button etc. After minimizing its FEWS GUI as a shade, the dominant robot immediately deletes the lock file **”fews_hindcast_robot_running”**, allowing the waiting robots to compete.

- When starting several robots simultaneously, it is possible that when you are still typing the commands in some xterms(e.g. on xterm 3) and the robot from earlier command (e.g. on xterm 1) has started its FEWS and started using the keyboard and mouse. This makes it impossible for you to finish typing. To avoid that, you may use **touch** command to manually create the empty lock file **”fews_hindcast_robot_running”** in your home directory first, forcing all the robots to wait. Then you may type all the commands at your own pace. After that, manually delete the blocking file, releasing the robots to race.
- After sometime the screen saver may kicks in due to lack of user activity to the computer. If the Robot tries to start another FEWS (e.g. the next one in the queue, or a re-run), its activity will be blocked by the screen saver. To bypass the problem, you may turn off the screen saver through the Desktop configuration. Alternatively, you may use a virtual desktop, like VNC SERVER and VNC VIEWER (explained next), which is the strongly suggested manner to run a hindcast.

[9] within this libs directory, there are two *.so files: libJXGrabKey.so and libVisionProxy.so. If you want to check that your environment variable LD_LIBRARY_PATH has been set correctly for Sikuli, run the commands ”**ldd libJXGrabKey.so**” or ”**ldd libVisionProxy.so**” to see if every library file needed by the two so files has been found.

3.3 Running in VNC SERVER/VNC VIEWER Virtual Desktop

There are many different kinds of virtual desktop. Here, we recommend VNC SERVER/VNC VIEWER, which are available by default on Linux computers. We have used and gained some experience with them, realizing some default values within the scripts need to be modified for better effect, as explained later. First, the following are some of the advantages of using a virtual desktop:

- When running inside the virtual desktop, the robots' keyboard/mouse action will be restricted inside the desktop and will not interfere with your keyboard/mouse activity on the same computer, but outside of the virtual desktop.

- Once you have finished typing all the commands in xterm(s) inside the virtual desktop, you may close the virtual desktop! And, you may even log off the Linux computer!! Being invisible, the robot(s), running inside the virtual desktop, will continue doing their activities as usual. You may log back to the virtual desktop from anywhere, any computer to inspect, and the screen saver is not an issue anymore.

To use the VNC SERVER/VNC VIEWER virtual desktop:[10]

1) Start VNC SERVER as first time, please type the command below and hit ENTER:

vncserver :11 (11 is just an example, you can pick up any one digit or two digit number)

You may be asked to create a password - which will be needed every time you open VNC VIEWER, so please don't lose it. Now, once VNC SERVER has been started for the first time, a hidden directory (**.vnc**) is created under your home directory. Inside it, there will be several items: the script xstartup, a *.log file and a *.pid file. Both the log file and the pid file name contain ":11", which is the port number we have used above to start the vnc server. Please close VNC SERVER now since we need to modify the script **xstartup** before we restart it. To shut down VNC SERVER:

vncserver -kill :11 (same port number as it was opened above)

Now inside **.vnc**, the *.pid file is gone. Since we later may let VNC SERVER run for weeks or months, the pid file inside **.vnc** can remind us on which machine and which port it is running. If there are two VNC SERVER running concurrently, there will be two *.pid files in existence. If you have never run VNC SERVER before, there won't be a .vnc file in your home directory. Once **.vnc** been created, it will be there forever.

2) Open xstartup with your favorite editor; the last line should be:

twm

which specifies that the window manager in the virtual desktop. This window manager is not compatible with the Robot, so please replace this line with the new line:

[10] man vncserver is useful

```
nohup kwin &
```

which starts a different window manager, kwin (which GNOME uses), in the virtual desktop. Adding “nohup” in front of it insures that the window manager won’t stop when we log off the computer!

Now, please re-start VNC SERVER as above and we are ready to run VNC VIEWER.[11]

3) Please type the command:

```
vncviewer :11 &
```

(same port number as vncserver is running)

A password window pops up. After entering the password which you created when running VNC SERVER the first time, the virtual desktop is up and running. If you have forgotten the password or you just don’t want to key in the password every time you start a vncviewer, you may use the command:

```
vncviewer –passwd ~/.vnc/passwd :11 &
```

man vncviewer provides helpful information.

[11] vncserver actually is a python script which starts Xvnc process and other processes. These processes won’t block you from exit the xterm when you log off the computer.

4 Additional Support

We hope you find this tool useful and welcome any suggestions for improvement. If you discover any issues, please use the established mechanism and write a fogbugz ticket.